

User manual of the landmark-based speech recognition toolkit

To accompany "Speech recognition based on phonetic features and acoustic landmarks", PhD thesis, University of Maryland, 2004

Amit Juneja

February 12, 2005

1 Synopsis

System Requirements:

A. SVM Light must be installed on the system B. Phoneme label files in TIMIT format must be available C. Frame-by-frame computed acoustic features in binary format (explained below) or HTK format D. Python 2.2 E. *nix (Unix, Linux, etc.) . It may run on Windows but I never tested it.

1. train_config.py

Usage: train_config.py <Config File>

This is the main executable for phonetic feature classification. It can (a) create files for use with MATLAB, SVM Light and LIBSVM by picking up acoustic parameters either by frame-by-frame basis or on the basis of landmarks, (b) train SVM classifiers (available only for SVM Light, and LIBSVM has to be run separately) while optimizing the kernel parameter and the penalty (bound on alphas) with different methods - minimum XiAlpha estimate of error, minimum number of support vectors, minimum cross-validation error, (c) do SVM classification on test files created by the code in a separate pass, (d) create histograms. SVMs for multiple phonetic features can be trained and tested at the same time. Please read the help in README.config for formatting the config file because this is the most crucial step.

2. print_landmarks.py

Usage: print_landmarks.py <Config File>

This will use the same config file as needed by `train_config.py` . It will create a landmark label file for each utterance in a list of utterances provided in the config file. The landmarks can be generated in one of the two ways: (a) using knowledge based acoustic measurements (b) using only the phoneme labels.

3. `collate_aps.py`

Usage: `collate_aps.py`

Combines two streams of acoustic parameters, for example, one stream of MFCCs and one stream of knowledge based acoustic measurements, by choosing only specified set of measurements from both the streams. It can also compute and append delta and acceleration coefficients for the selected measurements from both the streams. Binary and HTK format for both input and output are accepted. To create output files in HTK format, ESPS must be installed on the system, especially, the 'btosps' and 'featohk' commands must be available. To customize the command open the file `collate_aps.py` and follow the instructions.

4. `phn2lab.py`

Usage: `phn2lab.py` <phn file> <lab file>

Converts phn labels to ESPS format labels that can be displayed in xwaves.

5. `batch_phn2lab.py`

Usage: `batch_phn2lab.py` <phn file list> Converts label files in .phn format to ESPS .lab format given an input list of .phn files. It assumes that the input files have 3 character extension.

6. `findScalingParameters.py`

`findScalingParameters.py` <Config File>

Uses the same config file as in `train_config.py` to compute the scaling parameters for all of the acoustic measurements. This script must be run before running the `train_config.py` if scaled parameters are to be used.

7. File formats

Binary: This is plain binary format. Acoustic parameters are written frame-by-frame with each parameter in 'float'. For example, if there are 500 frames and 39 parameter per frame, then 39 parameters for the first

frame are written first, followed by the 39 parameters of the second frame, and so on. Note (1) each parameter is written in float (2) as far as this toolkit is concerned, linux and unix generated acoustic parameter files in binary format are not cross-compatible on these systems because the two systems use a different byte order.

2 Configuration files parameters

A number of values can be set in a config file that goes as input to the executables `train_config.py`. These are discussed here. Three examples of a config file are `config_broadclass_hie.py`, `config_mfc_hie.py` and `context_config.py` provided along with the scripts. The config variables are set in python format which has a very easy and obvious syntax. The code can be used for frame-based and landmark-based training and testing. Many experiments can be carried out by both frame-based and landmark based methods. Landmarks are computed by the system automatically for each phoneme by first converting a phoneme into a broad class label and then finding a set of landmarks for each broad class. The following landmarks are computed : Vowel (V) : [Vowel onset point (VOP), Peak] Sonorant consonant (SC - nasal or semivowel) : For postvocalic case, [Syllabic peak of previous vowel, SC onset, syllabic dip which is the mid point of the SC segment in this case], For prevocalic case, [syllabic dip which is the mid point of the SC segment in this case, SC offset (vowel onset), syllabic peak of the following vowel]. Intervocalic case: [Syllabic peak of previous vowel, SC onset, syllabic dip which is the mid point of the SC segment in this case, SC offset (vowel onset), syllabic peak of the following vowel] Stop (ST) : [Burst, Release] Fricative: [start frame, 1/4 frame, middle frame, 3/4 frame, end frame] Silence: [Silence start, silence end] The silence landmarks are useful for classification of the stop place features in postvocalic contexts.

The landmarks shown above for each broad class must be noted because this knowledge is essential for doing landmark-based experiments. In landmark based experiments, you need to specify where acoustic parameters are to be picked at. For example, if acoustic parameters 1,23,27 (this numbering is for the order in which the parameters are stored in parameter files starting with 1) are to be picked at Peak of the vowel, then the value of the Parameters variable below for such a class has to be set as [[], [1, 23, 27]] such that nothing is picked at the vowel onset point. In addition if a number of adjoining frames is to be used at Peak landmark then the value of Adjoins is set as [[], [-4, -2, 0, 2, 4]] and then the parameters [1, 23, 27] will be picked from (Peak - 4)th frame, (Peak - 2)nd frame and so on. For a particular classification, the current version of the code has a constraint that if the number of parameters at a landmark for a broad class are non-zero: then the number of parameters and the number of adjoins for that landmark must be the same as other non-zero ones. For example, if some parameters have to be picked from the VOP, then it should also have three parameters (considering above example) computed using the adjoins of size five, for example [-4, -1, 0, 1, 4]. Of course, the parameters and the adjoins may be

different.

A single config file can be used for a number of SVM classification experiments. In the config file you specify a list of SVM Light formatted data files, a list of model files names, indices of parameters to be extracted for each classification, etc. The i 'th element of each of these lists determine how the i 'th experiment is done.

1. Flags and values related to kinds of tasks and various inputs (labels and acoustic parameters)

outputDir

The full path of the directory containing the acoustic parameter files. A misnomer because this directory is more of an input.

labelsDir

The full path of the directory containing the label files in TIMIT format.

modelDir

The outout directory where model files and SVM Light formatted data files will be written.

filelist

Full path of a list of acoustic parameter files.

shuffleFilesFlag

If this is set to 1, the list of files will be shuffled before use

apFileExtLen

This an integer telling the length of extension of each acoustic parameter file. The code takes off this many number of characters and appends the label extension (refLabelExtension) to find the label file in the directory labelsDir .

refLabelExtension

The extension of the label file, for example, 'phn'

SkipDataCreationFlag

If this flag is test to 1, then no SVM formatted data files are created. This is used to only run the SVM Light, for example, to optimize the value of gamma or C .

SkipModelTrainingFlag

Setting this to 1 will skip model training. This can be used to (1) only create the SVM Light formatted data files so as to test with other toolkits such as LIBSVM of MATLAB externally, (2) create SVM Light formatted data files that can be used as validation files for SVM training in a separate pass.

SkipBinningFlag

Setting this to 1 will skip creation of bins for probabilistic modeling of SVM outputs. This not relevant for this version of teh code.

binaryClassificationFlag

If this flag is set to 1, SVMs will be run on the files in the array SvmInputFilesDevel

classificationType = 2

1: Non-Hierarchical 2: Hierarchical . Please ignore this flag in this version

of the toolkit. It is only relevant in the full-version

nBroadClasses

Please ignore this value in this version of the toolkit. It is only relevant in the full-version. Give it any value but do include it in the config file.

nBroadClassifiers = 4 # Not relevant for classification

Please ignore this value in this version of the toolkit. It is only relevant in the full-version. Give it any value but do include it in the config file.

nClasses

The number of SVMs . Not required but it can ease writing of certain variables in the config file that are same across all the SVMs to be trained. For example in python, a=['z']*5 will assign ['z', 'z', 'z', 'z', 'z'] to a .

selectiveTraining

The code allows for carrying out the designated tasks on a specified set of features instead of all the features. Even if config file is written for 20 SVMs (features), you can specify which features to analyze. For example, selectiveTraining = [0,3,5,6]

apDataFormat

0: binary, 1: HTK .

2. Values related to the names of SVM Light format files and model files to be created

SvmInputFiles

The names of SVM Light formatted files to be created. For example, SvmInputFiles = ['LightSonor', 'LightStops', 'LightSC', 'LightSilence']

SvmInputFilesDevel

The names of files used for validation. When optimizing a kernel related parameter, these files will be used to minimize the error on. For example, SvmInputFilesDevel = ['LightSonorDevel', 'LightStopsDevel', 'LightSCDevel', 'LightSilenceDevel']

modelFiles

The names of models. For example, modelFiles = ['rbf_model_sonor', 'rbf_model_stop', 'rbf_model_sc', 'rbf_model_sil']

3. Values and flags related to the parameters used in each classification

Parameters

The list of parameters to be used for each classification. For example, [[1, 2, 15, 16, 19], [4, 5, 17, 18], [8, 13, 14, 15, 16], [9, 4, 5, 6, 7]] where each list is a list of parameter for the corresponding index of model file, SVM data file, etc. These examples are good only for frame-based training. For landmark based testing, parameters are specified for each landmark as exemplified in the synopsis above. More examples can be found in the

config_mfc_hie.py (example file) file provided with the toolkit.

Doublets = [[]]*nClasses

Not tested in a while and better not to use. Assign Doublets = [[]]*nClasses to have the code ignore it.

Adjoins

The number of adjoining frames along with the current frame to be used for classification. For example, [[-4, -3, -2, -1, 0, 1], [-4, -3, -2, -1, 0, 1, 2, 3, 4], [-16, -12, -8, -4, 0, 4, 8, 12, 16, 20, 24], [-3, -2, -1, 0, 1, 2]]. For landmark-based training, adjoins have to be specified for each landmark as stated in the synopsis above.

numberOfParameters

The number of parameters per frame in each acoustic parameter file.

stepSize

The step size of the frames in milliseconds. Required for reading the labels.

classes_1

The +1 class members (phonemes/broad classes) from which the parameters are to be extracted. For example, classes_1 = [['V', 'SC', 'N'], ['ST', 'VST'], ['n', 'm', 'y', 'w', 'r', 'l', 'ng'], ['start-end', 'VB', 'epi', 'CL']]. See the file labels.py for the mapping used for phonemes to broad classes.

classes_2

The -1 class members (either phonemes or broad classes but not both in any classification) from which the parameters are to be extracted. For example, classes_2 = [['V', 'SC', 'N'], ['ST', 'VST'], ['n', 'm', 'y', 'w', 'r', 'l', 'ng'], ['start-end', 'VB', 'epi', 'CL']]. See the file labels.py for the mapping used for phonemes to broad classes.

useDurationFlag

A flag for each classification, for example, [0, 0, 0, 0]. A flag can take a value 1 only when the corresponding parameterExtractionStyles flag is set to 7 (landmark based training) .

specificDataFlags

If broad classes are used in classes_1 and classes_2 for any of the classification, set it to 0 otherwise set it to 1, for that classification.

parameterExtractionStyles

0: Frame based training, 1: IGNORE, not tested in a while, 7: landmark-based testing .

useDataBound

Setting this flag to 1 will use an upper bound on the number of samples extracted for each classification . The number is set by the values maxclass1 and maxclass2 explained below .

placeVoicingSpecifications

This selects the kind of landmark training for each classifier for which landmark training is chosen. For vowels the options are 'generic' (all vowels will be used), 'preSOnly' (vowels with no following sonorant consonant will be used and postSOnly (vowels with no preceding vowels will be used). For fricatives, the options are 'generic' (all fricatives), 'genericPreVocalic'

(fricatives before vowels and sonorant consonants), 'genericPostVocalic' (fricativated after vowels or sonorant consonants), 'genericIsolated' (fricatives with no adjoining sonorants). For sonorant consonants, the options are 'genericInterVocalicSC' (as the name suggests - note that there are five landmarks in this case), 'genericPreVocalicSC' (three landmarks) , 'genericPostVocalicSC' (three landmarks). For stops, the only valid option is 'genericPreVocalic'. The variable placeVoicingSpecifications will be removed in the forthcoming versions of the code and the framework will allow the user to specify any context.

init1

For frame-based training this is the list of numbers of initial frames to be extracted for each classifier. If for any classifier this value is set to non-zero, then only that number of initial frames will be used from classes.1 . The middleFlag1 will be ignored. For example, init1 = [0, 1, 0, 0] # Only relevant for frame-based training

init2

For frame-based training this is the list of numbers of initial frames to be extracted for each classifier. If for any classifier this value is set to non-zero, then only that number of initial frames will be used from classes.2 . The middleFlag2 will be ignored. For example, init2 = [0, 1, 0, 0] # Only relevant for frame-based training

delstart1

Delete an initial number of frames when picking frames for frame-based training from a label in classes.1. For example, delstart1 = [0, 0, 0, 0] . Only relevant for frame-based training. Ignored if a corresponding init1 value is set to non-zero .

delstart2

Delete an initial number of frames when picking frames for frame-based training from a label in classes.2. For example, delstart2 = [0, 0, 0, 0] . Only relevant for frame-based training. Ignored if a corresponding init2 value is set to non-zero.

delend1

Similar to delstart1 but for end frames.

delend2

Similar to delstart2 but for end frames.

contextFlag1

Specify the left and right context of eaach of the labels in classes.1. Only the phonemes/broad classes with the specified context will be used. If the ith element of the list contains 'left' or 'right' or both, then only those phonemes will be used that have the phonemes or broad classes specified in the context1 dictionary in the designated context. Currently this is only implemented for frame-based training. For landmark based training, use placeVoicingSpecification . The example file context_config.py shows an example of how to use context. If phonemes are specified in classes.1 and classes.2, then the context must also be phonemes, and the same for broad classes.

contextFlag2

Specify the left and right context of each of the labels in classes_2. Only the phonemes/broad classes with the specified context will be used. If the ith element of the list contains 'left' or 'right' or both, then only those phonemes will be used that have the phonemes or broad classes specified in the context2 dictionary in the designated context. Currently this is only implemented for frame-based training. For landmark based training, use placeVoicingSpecification . The example file context_config.py shows an example of how to use context. If phonemes are specified in classes_1 and classes_2, then the context must also be phonemes, and the same for broad classes.

context1

Specify the context . Relevant only if contextFlag1 is not empty. The element corresponding to the ith classifier is a dictionary in python format. For example, an element may be 'left': ['iy', 'ow'], 'right': ['k', 'g']. Many examples of using context are in the file context_config.py.

context2

Specify the context . Relevant only if contextFlag2 is not empty. The element corresponding to the ith classifier is a dictionary in python format. For example, an element may be 'left': ['iy', 'ow'], 'right': ['k', 'g']. Many examples of using context are in the file context_config.py.

randomSelectionParameter1

Instead of picking all frames pick frames randomly. For example, randomSelectionParameter1 = [0, 0, 0, 0]. This feature has not been tested in a while, so please prefer not to use it. # Only relevant for frame-based training

randomSelectionParameter2

Instead of picking all frames pick frames randomly. For example, randomSelectionParameter2 = [0, 0, 0, 0] . This feature has not been tested in a while, so please prefer not to use it. Only relevant for frame-based training

middleFlag1

Specify if only the frames from a middle portion of each label is to be used for training. 1: middle 1/3 segment, 2: middle 2/3 segment, 3: only the center frame. Example, middleFlag1 = [0, 0, 0, 0] # Only relevant for frame-based training

middleFlag2

Specify if only the frames from a middle portion of each label is to be used for training. 1: middle 1/3 segment, 2: middle 2/3 segment, 3: only the center frame. Example, middleFlag1 = [0, 0, 0, 0] # Only relevant for frame-based training

maxclass1

Maximum number of samples to be extracted for class +1. Example, maxclass1 = [20000, 5000, 20000, 20000] # Only relevant for frame-based training

maxclass2

Maximum number of samples to be extracted for class -1. Example, max-class2 = [20000, 5000, 20000, 20000] # Only relevant for frame-based training

4. SVM parameter settings

trainingFileStyle = 'Light'

Choice between 'Light' and MATLAB . If MATLAB is chosen then a binary file is written .

kernelType = [2, 2, 2, 2]

Same usage as SVM Light. 10 : Use known optimal gammas. Set the optimumGammaValues below For example, kernelType = [2, 2, 2, 2]

gammaValues

The set of values from which optimal is to be found. For example, gammaValues = [0.05, 0.01, 0.005, 0.001, 0.0005, 0.00001]

optimumGammaValues

If optimal gamma value is known for each or some of the classifications, set it here. For example, [0.01, 0.001, 0.001, 0.01] will set 0.01 as the optimal value for classification 0, 0.001 as optimal value for the classification of index 1 and so on.

cValuesArray = [0.05, 0.5, 1.0, 10]

Values of C from which best C is to be chosen. For example, cValuesArray = [0.05, 0.5, 1.0, 10]

flagCheckForDifferentC

If set to 0, default C found by SVM Light will be used .

svmMinCriterion

If set to 'numSV' the minimum number of support vectors will be used to get the optimum value of C as well as gamma . 'crossValidation' will cause the code to use validation across the files in SvmInputFilesDevel . The files in SvmInputFilesDevel need to be created in a separate run of the code by specifying the same names in the SvmInputFiles

BinsFilenames

The names of files that will contain the histogram binning information. For example, BinsFilenames= ['BinsSonor30RBF', 'BinsStops30RBF', 'BinsSC30RBF', 'BinsSilence30RBF'] . Binning is not relevant for this version of the code.

probabilityConversionMethod

Choice of 'bins' or 'trivial' . Trivial will use linear mapping from [-1,1] to [0,1]

binningBound

Bins will be constructed between -binningBound and +binningBound

5. Parameters for scaling

parameterScalingFlag

If this is set to 1, the parameters will be scaled by their empirical mean and variance. If set to 1, findScalingParameters.py must run before train_config .

scaleParameterFile

The full path of file to be created by findScalingParameters.py and to be read by train_config.py . For example, modelDir+'/'+'scaleFile'

scalingFactor

The value at which standard deviation of the scaled parameters is set.

scalingToBeSkippedFor

A list of indices of features for scaling is not to be used. For example, [0,4,5]

6. Parameter Addition Specifications : Deprecated: should be ignored but not deleted

addParametersFlag = 0

addDirectory = '/dept/isr/labs/nsl/scl/vol05/TIMIT_op/train'

temporalStepSize = 2.5

fileExts = ['aper.bin', 'per.bin', 'pitch.bin', 'soff.bin', 'son.bin']

channels = [1,1,1,1,1]

7. Ap specifications for landmark detection

useLandmarkApsFlags

Before landmark-based analysis is done, the code finds out the landmarks using the phoneme labels and optionally using knowledge based acoustic measurements. Landmarks are defined corresponding to broad classes vowel, fricative sonorant consonant (nasal or semivowel), silence and stop burst. If you want to use knowledge based measurements along with the phoneme labels for finding landmarks for any of the broad classes, set the corresponding flags as 1. For example, useLandmarkApsFlags = 'V':0, 'Fr':0, 'ST':1, 'SILENCE':0, 'SC':1 will cause the code to use measurements for the landmarks for ST and SC, and only the phoneme labels will be used to find the other landmarks. The parameters defined by the landmarkAps will be used.

landmarkAps

The index of the parameter for each of the measurements - onset, offset, totalEnergy, syllabicEnergy, sylEnergyFirstDiff - has to be set below. For

example, `landmarkAps = 'onset': 17, 'offset': 18, 'totalEnergy': 18, 'syllabicEnergy': 13, 'sylEnergyFirstDiff': 32` . Note that the first parameter is 1 and not zero. The maximum value of 'onset' parameter will be used to find stop burst. The maximum value of `totalEnergy` will be used to find the vowel landmark its minimum value will be used to find the dip of an intervocalic sonorant consonant. The maximum value of the `sylEnergyFirstDiff` will be used to find the SC offset (while moving from SC to vowel) and its minimum value will be used to find the SC onset (while moving from vowel to SC).